A Hybrid, Hierarchical Data Structure for Real-Time Terrain Visualization

Konstantin Baumann, Jürgen Döllner, Klaus Hinrichs, Oliver Kersting

Department of Computer Science, University of Münster
E-mail: {kostab,dollner,khh,kerstio}@uni-muenster.de

Abstract

The approximation tree is a hybrid, hierarchical data structure for real-time terrain visualization which represents both geometry data and texture data of a terrain in a hierarchical manner. This framework can integrate different multiresolution modeling techniques operating on different types of data sets such as TINs, regular grids, and non-regular grids. An approximation tree recursively aggregates terrain patches which reference geometry data and texture data. The rendering algorithm selects patches based on a geometric approximation error and a texture approximation error. Terrain shading and thematic texturing, which can be generated in a preprocessing step, improve the visual quality of level of detail models and eliminate the defects resulting from a Gouraud shaded geometric model since they do not depend on the current (probably reduced) geometry. The approximation tree can be implemented efficiently using object-oriented design principles. A case study for cartographic landscape visualization illustrates the use of approximation trees.

1 Introduction

Terrain models are discrete approximations of real terrains. Their specification is based on *geometric data* represented by a finite set of data points with associated elevation values (e.g., digital elevation models). Terrain models are used to relate *thematic geo-data* such as land use information or temperature fields with geometric data. Thus, they play an import role for managing and visualizing geo-data in geo-information-systems as well as in virtual reality applications (e.g., flight simulators).

Real-time terrain visualization requires multiresolution modeling in order to generate a terrain at different levels of detail for a given set of data points. Various hierarchical data structures have been developed for representing terrains, e.g., hierarchical TINs (de Floriani and Puppo [3]), R-trees for terrain data (Kofler et al. [9]), restricted quadtree triangulations (Pajarola [12]), and constrained Delaunay pyramids (Voigtmann et al. [16]). Each data structure supports a specific type of input data such as arbitrarily distributed data points for *triangulated irregular networks* (TINs) or regularly distributed data points for grids. In general, real-world data sets include

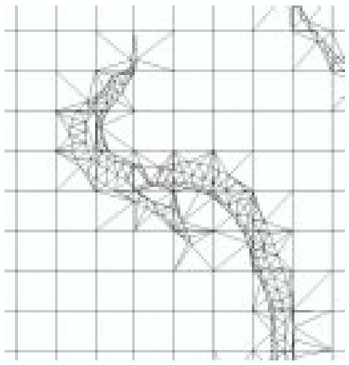

Figure 1: Terrain model consisting of a grid and additional microstructures provided as TINs.

data of different types. For example, a cartographic terrain model can include grid data describing the *digital elevation model* (DEM) and additional TINs describing structures at a finer resolution such as topographically complex or interesting terrain parts (e.g., river beds illustrated in figure 1). Multiresolution modeling can be extended to texture data as well. Textures are frequently used to visualize thematic data. For real-world data, large textures have to be processed which do not fit into graphics texture memory or even into main memory. Multiresolution terrain models for geometry and texture data are the prerequisites for interactive manipulation and real-time visualization of terrain data.

In the next section related approaches are discussed. Section 3 describes the definition of the approximation tree and section 4 the rendering algorithm. In section 5, the use of topographic and thematic texturing is illustrated. Section 6 sketches the class hierarchy of our object-oriented implementation. We conclude the paper by describing a case study realized in cartography.

2 Related Work

A large number of multiresolution modeling techniques for arbitrary polygonal meshes and, in particular, for digital terrains have been developed in the past. Hierarchical triangulations based on TINs have been applied to generate multiresolution models which can be used by level of detail algorithms (de Berg and Dobrint [2], de Floriani and Puppo [3], Gross et al. [8], and Voigtmann et al. [16]). Regular grids have been used for multiresolution modeling (Falby et al. [6]) and for real-time, continuous LOD rendering (Lindstrom et al. [10], and Pajarola [12]). These approaches suffer from the following limitations:

- Most terrain modeling techniques are based on one single multiresolution geometric model (e.g., hierarchical TINs) and cannot integrate different types of terrain data into one hierarchical terrain model. However, for many applications it is necessary to operate on hybrid data sets because they are bound tightly to numerical models or simulation models and therefore should not be standardized (i.e., converted) into a uniform representation.
- Only few terrain modeling techniques (e.g., Fowler and Little [7], and Douglas [5]) support a level of detail mechanism which adapts itself according to the data semantics. For example, a patch of a multiresolution terrain model which contains a river bed should use a TIN representation at a fine-grained level, whereas the patch could use a regular grid at a coarse-grained level.
- Terrain modeling techniques are optimized for rendering speed - sometimes at the cost of visual quality with respect to topographic features or thematic data visualization. For example, Gouraud shading based on a coarse triangulation results in a misleading shading because it is based on the triangle vertex normals without taking interior data points or topographic features into consideration (this is a problem even for high resolution triangulations).
- Terrain modeling techniques are mainly designed for representing geometric terrain data. However, thematic data plays an important role in real applications of terrain visualization (e.g., in a GIS), because the terrain model forms the basis for the interactive manipulation of terrain-related geo-data (e.g., in landscape planing). Therefore, multiresolution texture models are as important as multiresolution geometry models.

Our primary goal is *not* to define a new technique for building levels of detail. Instead we propose a generic data structure which allows us to integrate several multiresolution techniques, e.g., continuous LOD of height fields (Lindstrom et al. [10]) or restricted quadtree triangulations (Pajarola [12]), in a single multiresolution model. Furthermore, the proposed data structure can be extended to multiresolution modeling for texture data.

3 Approximation Trees

In this section, we define the approximation tree data structure and show how it can be constructed recursively. The general idea is to start with a terrain data set given as a set of data points in a plane together with an elevation function, and to recursively subdivide the terrain according to a given scheme.

3.1 Definition: Approximation Tree

Let $P = \{p_1, ..., p_m\}$ be a set of data points in the xy-plane. Let a_{min} and a_{max} be the extremal points of the axis parallel bounding box of P. Let $T = (P, h_T)$ be a terrain model given by a set of data points P and an elevation function h_T. The domain of T is defined as $D(T) := \{(x, y)|x_{a_{min}} \leq x \leq x_{a_{max}}, y_{a_{min}} \leq y \leq y_{a_{max}}\}$. The elevation function $h_T : D(T) \rightarrow R$ defined on $D(T)$ calculates elevation values for points of $D(T)$ by interpolation using the data points of P.

Let $Q := \{q_{u,v}|1 \leq u \leq m, 1 \leq v \leq n\}$ be a texture consisting of the $m \times n$ texels $q_{u,v}$; b_{min} and b_{max} denote the extremal points of the axis parallel bounding box used to geo-reference this texture. In analogy, the domain $D(Q)$ of the texture Q is defined by $D(Q) := \{(x, y)|x_{b_{min}} \leq x \leq x_{b_{max}}, y_{b_{min}} \leq y \leq y_{b_{max}}\}$.

An *approximation tree* $A_{s,d}(T)$ for a terrain T consists of a set of nodes, where each node N represents a rectangular region $D_N \subset D(T)$ and approximates the terrain in that region by an approximating terrain $T_N = (P_N, h_{T_N})$, called a *geometry patch*. P_N consists of at most s data points (i.e., $|P_N| \leq s$). The bounding box of P_N is the domain D_N. The way the node calculates the data points P_N depends on the approximation strategy adopted by the node. For example, a grid-based node will select evenly spaced points from a grid data set, whereas a TIN-based node will select points from an arbitrary data set based on an error criterion.

Each node N can have at most d child nodes. The child nodes are constructed as follows: Let T_N be the approximating terrain of a node N having the domain $D_N \subset D(T)$. Define the *geometric approximation error* $\epsilon_T(N)$ as the maximal vertical distance between the terrains T and T_N, i.e., $\epsilon_T(N) := \max_{p \in D_N} |h_T(p) - h_{T_N}(p)|$. If the error between T and T_N exceeds a certain threshold $\varepsilon \geq 0$, i.e., $\epsilon_T(N) > \varepsilon$, then the domain of N is subdivided into a set of at most d rectangular subdomains such that $D_N = \bigcup D_{N_i}$ with $D_{N_i} \cap D_{N_j} = \emptyset, i \neq j, 1 \leq i, j \leq d$. For each subdomain D_{N_i}, a node N_i will be constructed which approximates the terrain in that subdomain, and will be added as child node to the parent node N. The domain of the root node of the approximation tree is $D(T)$ covering the whole domain of the terrain T.

An approximation tree node N contains a texture patch Q_N with $D_N \subset D(Q_N)$, i.e., the domain of the texture in terms of geo-coordinates covers the domain of the geometry patch. Q_N approximates the texture Q in a suitable way such that it can be used for the geometry patch N.

The approximation tree generalizes several hierarchical terrain representations, for example:

- *Quadtrees*: the terrain $T_{grid} = (P_{grid}, h_{T_{grid}})$ defined by a regular grid of $(2^m + 1) \times (2^n + 1)$ data points P_{grid} can be represented in a quadtree-like structure by an approximation tree $A_{(4+1)^2,4}(T_{grid})$ in which each node represents one quad of the parent's domain. Each

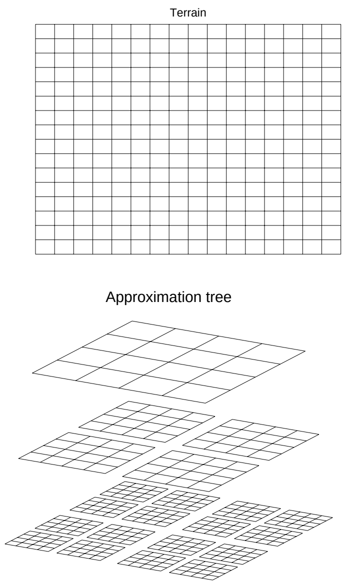

Figure 2: Terrain based on a regular grid represented by an approximation tree $A_{(4+1)^2,4}(T_{grid})$.

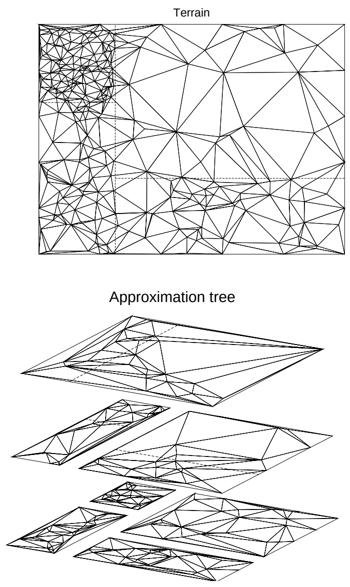

Figure 3: Terrain based on an arbitrarily distributed point represented by an approximation tree $A_{25,2}(T_{arb})$.

quad is approximated by $(4+1) \times (4+1)$ data points, structured as a regular grid (see figure 2).

- *Two-dimensional binary trees*: the terrain $T_{arb} = (P_{arb}, h_{T_{arb}})$ defined by a set of (arbitrarily distributed) data points P_{arb} can be represented in a k-d tree-like structure by an approximation tree $A_{25,2}(T_{arb})$ in which the approximating terrain in a node contains up to 25 vertices. The domain of each non-leaf node is split into two subdomains (see figure 3).

3.2 Integrating Different Approximation Strategies

Applications operating on hybrid data sets can benefit from different approximation strategies within a single approximation tree. A node of an approximation tree specifies an *approximation strategy*, i.e., the algorithm for subdividing the terrain domain and the strategy for selecting appropriate data points. In our sample framework, we have implemented the following strategies:

Regular Grids: The approximation can be based on selecting points of a regular grid by specifying the equidistant rows and columns the node actually considers. The original data set has to be stored only once, and can be referenced by multiple nodes.
The memory requirements are minimized because only the step sizes for rows and columns have to be stored in the node. Due to the regular structure, in general the approximation error is larger compared to other approximation strategies because we cannot adapt the grid to the characteristics of the underlying terrain.

TINs: A TIN-based approximation strategy inserts those points into an initially empty TIN which cause the largest approximation errors. We stop to insert points if the approximation error becomes zero or if the TIN reaches the maximum number s of points (see van Kreveld [15]).
This strategy generates the best results because the TIN adapts itself to the individual characteristics of a terrain. The memory requirements are high because of the irregularly distributed points, i.e., the vertices of the triangles have to be stored explicitly. An optimized strategy can compress the representation by building triangle strips (see, for example, Speckmann and Snoeyink [14]).

Non-Regular Grids: If a grid data set is given, we can select rows and columns which are not regularly spaced in order to provide a higher resolution in those areas which the application or the visualization considers to be important. We insert into an initially empty non-regular grid the row and column of that point which causes the largest approximation error. Like for the TIN strategy, we stop if the approximation error becomes zero or if the non-regular grid reaches the maximum number s of points. In the non-regular grid, only the intersection points of a row-column pair are used as points of the approximating terrain (see figure 4).
This strategy combines the advantages of TIN-based and grid-based approximation strategies for grid-based data because it has low memory requirements and adapts itself to terrain characteristics. The approximation error ranges roughly between the approximation error of the TIN and regular grid, while the memory consumption is slightly higher compared to the regular grid approach.

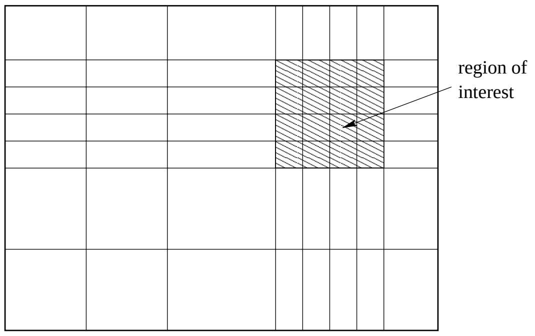

Figure 4: A non-regular grid approximation with a region of special interest.

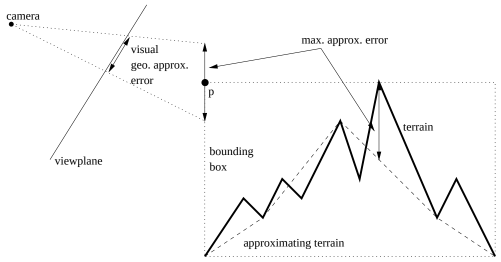

Figure 5: Calculating the visual geometric approximation error α.

3.3 Multiresolution Models for Textures in Approximation Trees

Each node N of an approximation tree $A_{s,d}(T)$ represents a part of the original terrain T by an approximating terrain T_N which will be rendered as a single graphics object. Texturing is the main mechanism to relate terrain geometry with thematic information (e.g., road maps, land use, etc.). Hence, texture data tends to be at least as extensive as geometry data in GIS applications. Multiresolution modeling for texture data should be tightly integrated with multiresolution modeling for geometry data.

Therefore each geometry patch is associated with a texture patch. Texture patches are built based on geometry patches. A texture patch can contain the actual texture data or can reference the texture data of another texture patch. A texture patch might reference only a subregion of the texture data.

Since rendering systems and graphics hardware impose certain constraints on the format and size of textures, textures have to be scaled and split into parts which satisfy these texture constraints (e.g., OpenGL requires textures with a size of a power of 2 and limits the maximum size of textures). Another constraint of most rendering systems is the fact that only a single texture can be applied to a single graphics object at the same time. Therefore, the domain $D(Q_N)$ of an approximating texture Q_N stored in a node N must cover the whole domain D_N.

We use the following strategy to generate LODs for textures: First, build a so called texture pyramid (or mipmap, Williams [17]) for the given texture data (which should not be confused with the texture mipmapping provided by some 3D rendering systems). It is conceptually a list of textures with the following property: each texture is a scaled-down version of its predecessor texture in the list. The first element in the list is the originally given texture, and the last element is a texture which satisfies all texture constraints of the underlying rendering system. All textures in a texture pyramid have the same domain as the original texture.

Each geometry patch T_N stored in a node N references a part of a texture stored in the texture pyramid (i.e., a *subtexture*). This part must satisfy the texture constraints of the underlying rendering system and is computed in the following way: Choose the first texture Q of the texture pyramid so that the size of the subtexture Q' covering *exactly* the domain D_N is less than the maximum texture size allowed by the texture constraints. Then determine a subtexture Q_N of Q that contains Q', i.e., covering *at least* the domain D_N, and satisfies all texture constraints imposed by the underlying rendering system. Q_N is the texture patch suitable for texturing the geometry patch T_N with the highest possible texture resolution. Note that the texture patch Q_N can also be used for texturing the descending patches of node N, since it covers the whole domain D_N of T_N, but probably with a non-optimal texture resolution.

This texturing technique can be extended to use different textures for different levels of detail. For example, a satellite image used as texture data for a terrain could be refined by high-resolution, detailed textures (e.g., city maps) which are used for nodes below a certain level.

If enough texture memory is available, we can also construct hardware accelerated mipmaps for each texture patch (see Woo et al. [18]). These hardware accelerated mipmaps are independent from the texture pyramid described above. They are used by the rendering system to speed up the texturing process and to improve the visual texture quality, but involve a slight memory overhead.

4 Rendering of Approximation Trees

The rendering of approximation trees is generic with respect to different approximation and subdivision strategies. The selection of nodes of an approximation tree is based on the view frustum culling and the visual approximation errors.

4.1 Visual Approximation Errors

The visual approximation errors measure the quality of an approximating terrain for the current camera settings. The quality depends on the geometric approximation error and the texture approximation error.

Let $A_{s,d}(T)$ be an approximation tree, N one of its nodes which contains the approximating terrain T_N, and B_N its three-dimensional axis parallel bounding box. Then, the visual geometric approximation error α_N and the visual texture approximation error γ_N are calculated as follows:

1. Test whether B_N intersects the current view frustum of the camera.

2. If B_N does not intersect, the node N and all its child nodes are not visible and can be skipped.

3. If B_N does intersect, find the point p of the bounding box B_N which is nearest to the camera position.

   (a) Construct a line segment along the z-axis (the direction of elevation) of length $\epsilon_T(N)$ centered at p. Project that segment onto the view plane. The *visual geometric approximation error* α_N is the length of the projected line segment measured in pixels (see figure 5).

   (b) Determine the width w and height h of a texel of the texture patch Q_N of the node N in the coordinate system of T_N. Construct a rectangle with the extension $w \times h$ parallel to the xy-plane centered at p. Project that rectangle onto the view plane. The *visual texture approximation error* γ_N is the area of the projected rectangle measured in pixels. γ_N specifies how much the area of the projected rectangle differs from a single pixel.

The visual approximation errors α_N and γ_N can be used as a measure for the visual quality of the corresponding patch N, because if we actually render the approximating terrain T_N, then it is guaranteed that each of the pixels of T_N differs by at most α_N pixels from the original terrain T. Furthermore, if the visual texture approximation error is below an appropriate threshold (≥ 1), we can guarantee that the texels are sufficiently "dense" for the approximating terrain.

4.2 Rendering Process

The terrain T represented by an approximation tree $A_{s,d}(T)$ can be rendered recursively. We start with the root node N_0 of $A_{s,d}(T)$. For a node N and its approximating terrain T_N, the rendering process is defined as follows:

Check whether the bounding box B_N of N intersects the current view frustum.

- If the bounding box does not intersect the view frustum, we skip the node and all its child nodes.

- If the bounding box intersects the view frustum and N has child nodes N_i, then we calculate the visual approximation errors α_N and γ_N. If α_N is larger than a user-defined geometric approximation threshold or if γ_N is larger than a user-defined texture approximation threshold, then render the child nodes N_i of N recursively. Otherwise, we can render T_N using the texture Q_N, and then stop rendering this branch of the tree.

- If the bounding box intersects the view frustum and N is a leaf node, then render T_N using the texture Q_N.

The rendering process must also be aware of gaps which may result at the border of two geometry patches. Standard techniques can be used to fill the gaps, e.g., inserting vertical walls.

Since a geometry patch T_N can also be textured with one of the texture patches of the ancestor nodes of N, the total number of textures used for rendering a frame can be reduced as follows: If a geometry patch T_N of a node N has been selected for rendering, T_N can also be textured with a texture patch Q' that belongs to an ancestor of node N, has a visual approximation error γ' less than or equal to the user-defined texture approximation threshold, and is closest to the root of the approximation tree. With this strategy, a texture patch serves for many geometry patches simultaneously which causes less texture switches and improves the rendering speed.

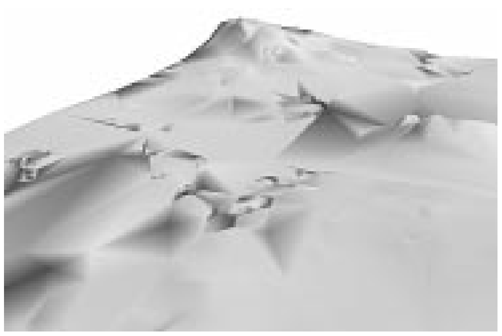

Figure 6: Sample terrain consisting of approximately 5000 triangles rendered by standard OpenGL Gouraud shading.

5 Topographic and Thematic Texturing

In geo-applications, the geometric representation of a terrain represents the basis for visualizing topographic and thematic data. Approximation trees are able to provide multiresolution models for both categories of data, geometry and texture data. The texturing plays a central role because it encodes both topographic and thematic information.

5.1 Topographic Texturing

The terrain texture includes shading information, i.e., luminance values which depend on the surface properties, the surface geometry, and the light sources. We propose to compute an explicit topographic texture (i.e., we do not apply a triangle-based Gouraud shading for the geometric model) due to the following observations:

- If Gouraud shading would be used, the vertex normals would determine the shading of a triangle. For large triangles, a lot of detail information would be lost, because the interior pixels of a triangle are shaded by interpolating the colors linearly between the vertices. Since mainly the shading transmits topographic features to the viewer, the visual quality can be increased dramatically if a topographic texture is used (compare figures 6 and 7). Then, even for large triangles detailed shading information is visible. Topographic textures can be implemented by luminance textures, available, for example, in OpenGL.

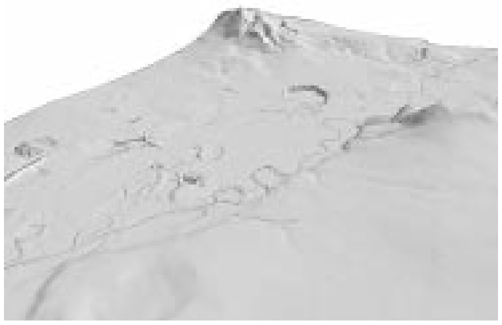

Figure 7: Sample terrain from figure 6 rendered with a topographic texture.

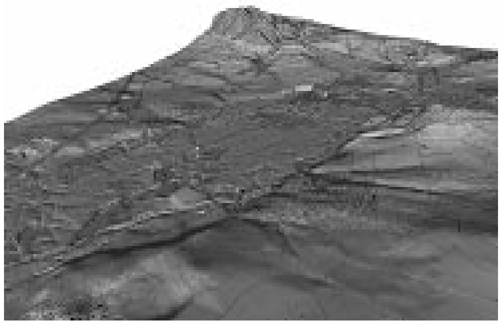

Figure 8: Sample terrain from figure 7 rendered with a combined topographic and thematic texture.

- If light sources and terrain geometry data are static, the diffuse lighting does not depend on the camera position. Thus, we can precompute the topographic texture. If the light sources or the terrain geometry change, we can recompute the topographic texture automatically.

- If we calculate a topographic texture, we are not limited to the lighting and shading models provided by 3D rendering systems such as OpenGL. We can use, for example, the Phong lighting model which achieves a better visual quality. In addition, shadows could be computed for a landscape.

- If morphological features such as ridges and hollows should be visualized, the computation of the topographic texture can be customized by an application. For example, procedural shaders could be applied. Those features could hardly be visualized using standard Gouraud shading unless they would have been geometrically modeled.

Topographic textures lead to a better perception of a terrain's morphology because humans perceive a terrain not only by the geometric silhouette but mainly by the surface shading. Hence, geometry changes can be hidden from the user by a static topographic texture which is in general independent from the current geometry level of detail.

In order to speed up the rendering performance, the topographic texture (eventually combined with a thematic texture) can be applied to the geometry patches using the decal mode (see, for example, Woo et al. [18]), i.e., the texture is mapped onto the geometry without lighting calculations or being modulated with the geometry surface color, saving expensive per-vertex calculations.

5.2 Thematic Texturing

Thematic texturing is responsible for mapping application relevant information onto the terrain geometry, for example, land use information or cartographic symbols. Thematic texturing is a separate process, which could combine different textures (even procedural textures) into a current thematic texture. The combination of thematic textures depends on the geo-information to be visualized. For example, a cartographic symbol texture could be overlaid by a land use texture. During rendering, the current thematic texture will be blended with the topographic texture. The resulting texture is used by graphics hardware for the current terrain geometry (see figure 8).

5.3 Offscreen Texture Generation

A standard topographic texture can be generated using offscreen rendering in OpenGL together with Gouraud shading. We use an orthographic projection of the given terrain with a geometry resolution appropriate for the texture resolution, provide light sources and vertex normals for the triangles, and capture the resulting image as a texture. This way, we can generate the topographic texture automatically and possibly accelerated by graphics hardware, and further refine the result in a post-processing step. The same approach can be taken to generate thematic textures, e.g., applying thematic functions to vertex colors, and capturing the resulting orthographic projection of the terrain. The accumulation buffer of OpenGL can be used to blend different textures, and the recently introduced imaging features of OpenGL 1.2 (see Segal and Akeley [13]) such as the convolution filters, the color transformation matrix and blending enhancements can be used to implement and accelerate the post-processing of both types of textures. For the near future, we can even expect hardware with accelerated procedural shading capabilities (Olano and Lastra [11]).

6 Object-Oriented Implementation

The approximation tree can be implemented in an object-oriented fashion taking advantage of inheritance and polymorphism. An approximation tree object is represented by a collection of geometry data objects, geometry patch objects, texture data objects, and texture patch objects. The class diagram is illustrated in figure 9.

- A geometry data object contains geometric data, e.g., a DEM containing 2D points and their elevation values. Actual geometry data objects belong to a spe-

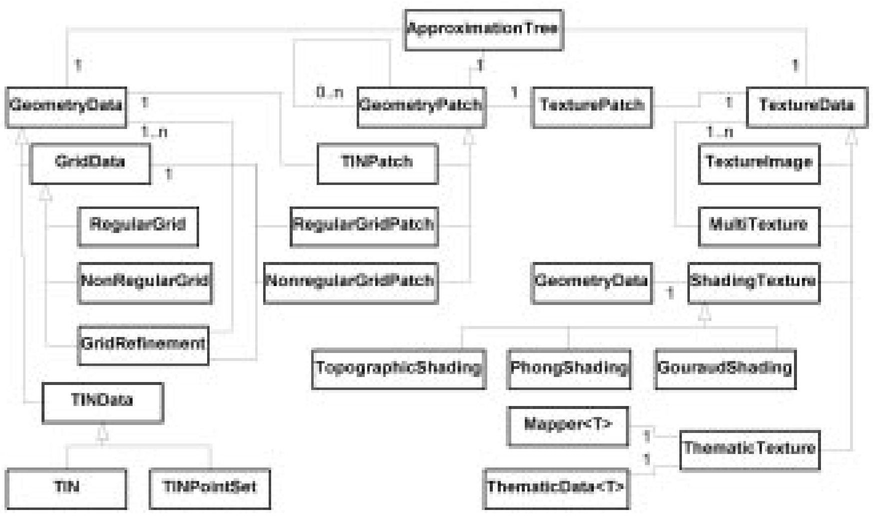

Figure 9: Class diagram for the approximation tree implementation.

cialized geometry data class, e.g., precomputed TINs (`TIN`), TINs given by a set of arbitrarily distributed points (`TINPointSet`), regular grids (`Grid`), non-regular grids (`NonRegularGrid`), and grids which contain additional fine-grained substructures such as TINs (`GridRefinement`). Their common base class (`GeometryData`) defines generic methods all subclasses of geometry data have to provide, e.g., access to vertices and normals. New types of geometry data can be integrated by subclassing this base class.

- A geometry patch object represents an approximating terrain at a certain level of detail. Geometry patch objects are recursively aggregated. At the moment, we have integrated two geometry patch types, TIN-based and grid-based patches. TIN-based patch objects (`TINPatch`) can be constructed from any type of geometry data objects, whereas grid-patch objects (`GridPatch, NonRegularGridPatch`) can be constructed only for grid data objects.

- A texture data object (`TextureData`) contains the texture data given by RGBA values (red, green, blue and alpha coefficients). At the moment, we have integrated the following texture types: A shading texture object (`ShadingTexture`) is associated with a geometry data object and represents its surface shading. Different lighting models such as Gouraud shading, Phong shading, and topographic shading are implemented by subclasses. A `ThematicTexture` object is associated with a thematic data object and a mapper object which transforms thematic data into RGBA values. A `MultiTexture` object blends two or more texture objects together into a new texture data object.

- A texture patch object (`TexturePatch`) encapsulates an image used for texturing an approximating terrain represented by the associated geometry patch object. Texture patch objects are recursively aggregated. A texture patch object knows its geo-referencing bounding box and sets up the appropriate texture transformations automatically.

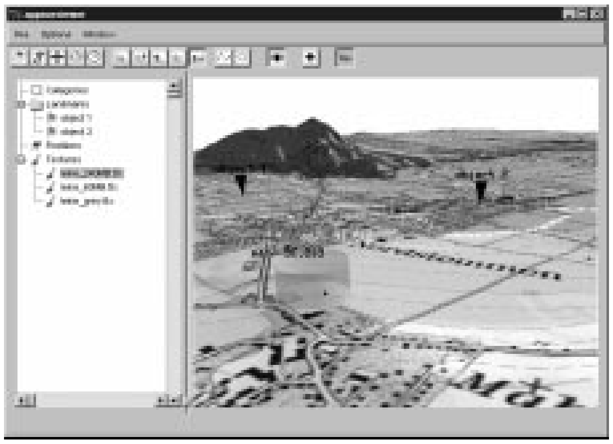

Figure 10: Snapshot of a cartographic landscape.

The object-oriented design facilitates the integration of new level of detail strategies as well as new methods for topographic shading and thematic texturing.

7 Case Study: Cartographic Visualization

The approximation tree concept has been used as a basic data structure for a cartographic visualization environment (Buziek and Döllner [1]). The thematic and geometric data sets include a coarse-grained terrain grid with given microstructures provided as precomputed TINs which have been modeled explicitly (see figure 10).

Thematic information includes different cartographic objects (road map, land use information, flooding in planing areas etc.) provided as textures. This interactive visualization environment is used for landscape planing tasks. The user can navigate through the terrain interactively and explore the different thematic data sets.

Our implementation of the approximation tree achieves interactive framerates of 15-20 frames per second on a low-cost PC platform (200 MHz Intel PentiumPro, 128 MB main memory, Diamond Fire GL 1000 with 4 MB graphics memory, Windows NT 4.0, window size 640×480 pixels). For these measurements we used a hybrid data set of a regular 641×321 grid and 20.652 microstructures (e.g., river beds) provided as TINs. The resulting terrain model consists of about 500.000 triangles. We used a 6144×12288 pixel high-resolution thematic texture combined with a topographic shading texture of the same size, resulting in a 240 MB RGB texture which was mapped to main memory by the memory-mapped-file mechanism provided by the operating system. For our tests, we have chosen a threshold of 4 for the visual geometric approximation error α and a threshold of 2 for the visual texture approximation error γ, which gives near optimal visual results.

8 Conclusions

The approximation tree represents a hybrid, hierarchical data structure which allows for integrating different level of detail strategies and data sets. Thus, it is well suited for inhomogeneous real-world data sets found in geo-sciences and cartography. For approximation trees, application data has not to be converted into a different format, and the original data semantics are not lost.

The approximation tree provides multiresolution models for both terrain geometry and terrain texture. For both, geometry and texture, different levels of detail are automatically constructed. The rendering process considers both screen-space geometric errors and texture errors. Thus, it reflects the need for visual quality and geometric correctness.

The anticipated topographic shading and the capability of blending thematic and topographic textures improves the visual quality dramatically because it outwits human perception by providing detailed shading information for a varying geometry. In addition, this terrain texturing approach allows for using non-standard shading algorithms not found in 3D real-time rendering systems, and takes advantage of accelerated texture mapping found in today's graphics hardware. The precomputation of the textures also improves the rendering speed.

The approximation tree can be implemented efficiently on top of modern, real-time 3D rendering systems such as OpenGL or Direct3D. For example, advanced and recently integrated OpenGL features, such as convolution filtering, color space transformations, and multilayered textures, can be exploited by our implementation.

This work forms part of an on-going project on a visualization environment for interactive cartography based on the 3D visualization system MAM/VRS (Döllner [4]). For more information, see `http://wwwmath.uni-muenster.de/~mam`.

Acknowledgments: We would like to thank Gerd Buziek for his collaboration on cartographic visualization. This work has been supported by the Ministry of Science and Research of North-Rhine-Westphalia, Germany.

References

[1] G. Buziek, J. Döllner. Concept and Implementation of an Interactive Cartographic Virtual Reality System for Landscape Planning. *Proceedings of International Association of Cartography*, 1999, to appear.

[2] M. de Berg, K. Dobrindt. On Levels of Detail in Terrains. *Proceedings of 11th ACM Symposium on Computational Geometry*, C26-C27, 1995.

[3] L. de Floriani, E. Puppo. Hierarchical Triangulation for Multiresolution Surface Description. ACM *Transactions on Graphics*, 14(4):363-411, 1995.

[4] J. Döllner, K. Hinrichs. Object-Oriented 3D Modelling, Animation and Interaction. *The Journal of Visualization and Computer Animation*, 8(1):33-64, 1997.

[5] D.H. Douglas. Experiments to locate ridges and channels to create a new type of Digital Elevation Model. *Cartographica*, 23(4):29-61, 1986.

[6] J.S. Falby, M.J. Zyda, D.R. Pratt, R.L. Mackey. NPSNET: Hierarchical Data Structures for Real-Time Three-Dimensional Visual Simulation. *Computers & Graphics*, 17(1):65-69, 1993.

[7] R.J. Fowler, J.J. Little. Automatic extraction of irregular network digital terrain models. *Computer Graphics (SIGGRAPH '79 Proceedings)*, 13(2):199-207, 1979.

[8] M.H. Gross, R. Gatti, O. Staadt. Fast Multiresolution Surface Meshing. *Proceedings Visualization '95*, 135-142, 1995.

[9] M. Kofler, M. Gervautz, M. Gruber. The Styria Flyover - LOD management for huge textured terrain models. *Proceedings Computer Graphics International 1998*, 444-454, 1998.

[10] O. Lindstrom, D. Koller, W. Ribarsky, L.F. Hodges, N. Faust, G.A. Turner. Real-time, Continuous Level of Detail Rendering of Height Fields. *Computer Graphics (SIGGRAPH '96 Proceedings)*, 109-118, 1996.

[11] M. Olano, A. Lastra. A Shading Language on Graphics Hardware: The PixelFlow Shading System. *Computer Graphics (SIGGRAPH '98 Proceedings)*, 159-168, 1998.

[12] R. Pajarola. Large Scale Terrain Visualization Using The Restricted Quadtree Triangulation. *Proceedings Visualization '98*, 19-26, 1998.

[13] M. Segal, K. Akeley. The OpenGL Graphics System: A Specification (Version 1.2). *Silicon Graphics, Inc.*, 1998.

[14] B. Speckmann, J. Snoeyink. Easy triangle strips for TIN terrain models. *Canadian Conference on Computational Geometry*, 239-244, 1997.

[15] M. van Kreveld. Digital Elevation Models and TIN Algorithms. In: M. van Kreveld, J. Nievergelt, T. Roos, P. Widmayer (eds.), *Algorithmic Foundations of Geographic Information Systems*, Lecture Notes Computer Science, 1340:37-78, Springer-Verlag, 1997.

[16] A. Voigtmann, L. Becker, K. Hinrichs. A Hierarchical Model for Multiresolution Surface Reconstruction. *Graphical Models and Image Processing*, 59:333-348, 1997.

[17] L. Williams. Pyramidal parametrics. *Computer Graphics (SIGGRAPH '83 Proceedings)*, 17(3):1-11, 1983.

[18] M. Woo, J. Neider, T. Davis. OpenGL Programming Guide (2nd Edition). *Addision-Wesley*, 1997.